Paradigmas de la Programación – Recuperatorio del Primer Parcial

19 de Junio de 2018

Apellido y Nombre: _____

1. **[20 pt.]** Explique por qué se usa "`X::f()`" en el siguiente fragmento de código en C++.

```
class X {
  public: virtual void f() {

  }
};

class Y : public X {
  public: virtual void f() {

  }
};

class Z : public Y {
  public: virtual void f() {
    X::f();
  }
};
```

2. **[20 pt.]** Estos dos fragmentos de código tienen la misma semántica, pero en uno hay inversión de control e inyección de dependencia, y en el otro no. Explique cuál es cuál. En el que tiene inversión de control, identifique el hot spot donde se puede parametrizar la dependencia que se inyecta en lugar de que esté incrustada en el código.

```
Class Engine{...}

Class Car{
  private Engine e;
  Car(){
    e= new Engine();
  }
}
```

```
Class Car{
  private Engine e;

  Car(Engine e){
    this.e = e;
  }
}
```

3. **[10 pt.]** La siguiente expresión está mal tipada:

```
f (a, b) = a > b || a
```

Muestre con el árbol para inferencia de tipos dónde se encuentra el conflicto y en qué consiste. Cómo podría resolver este conflicto un lenguaje de tipado fuerte? y uno de tipado débil?

4. **[20 pt.]** En el siguiente programa, diagrame el estado en el que se encuentra la pila de ejecución al terminar de ejecutar las líneas señaladas en el texto del programa.

```
a = 1
b = 2
c = 3

function test()

  local function g()
    local a = 2
    c = a + 4  ------
  end

  local function f()
    local c = 5
    b = 4      ------
    g()
  end
             ------
  f()
end
             ------
test()
```

5. **[10 pt.]** Lea el siguiente texto y explique con sus propias palabras el nivel de visibilidad `package` en Java, y relaciónelo con los niveles de visibilidad `private` y `protected`.

> *Access level modifiers determine whether other classes can use a particular field or invoke a particular method. There are two levels of access control:*
>
> - *At the top level—public, or package-private (no explicit modifier).*
> - *At the member level—public, private, protected, or package-private (no explicit modifier).*
>
> *A class may be declared with the modifier public, in which case that class is visible to all classes everywhere. If a class has no modifier (the default, also known as package-private), it is visible only within its own package (packages are named groups of related classes).*
>
> *At the member level, you can also use the public modifier or no modifier (package-private) just as with top-level classes, and with the same meaning. For members, there are two additional access modifiers: private and protected. The private modifier specifies that the member can only be accessed in its own class. The protected modifier specifies that the member can only be accessed within its own package (as with package-private) and, in addition, by a subclass of its class in another package.*

6. **[20 pt.]** De estos dos fragmentos de códigos, cuál es un ejemplo de polimorfismo y cuál es un ejemplo de sobrecarga? Justifique su respuesta.

```
public class Animal{
   public void sound(){
      System.out.println("Animal is making a sound");
   }
}

class Horse extends Animal{
    @Override
    public void sound(){
        System.out.println("Neigh");
    }
}

public class Cat extends Animal{
    @Override
    public void sound(){
        System.out.println("Meow");
    }
}
```

```
class Examp
{
    void demo (int a)
    {
       System.out.println ("a: " + a);
    }
    void demo (int a, int b)
    {
       System.out.println ("a and b: " + a + "," + b);
    }
    double demo(double a) {
       System.out.println("double a: " + a);
       return a*a;
    }
}
class MethodExamping
{
    public static void main (String args [])
    {
        Examp Obj = new Examp();
        double result;
        Obj .demo(10);
        Obj .demo(10, 20);
        result = Obj .demo(5.5);
        System.out.println("O/P : " + result);
    }
}
```